Mitsunori Hirata, Yuichi Sato, Fumio Nagashima, and Tsugito Maruyama
Fujitsu Laboratories Limited
1015 Kamikodanaka, Nakahara-ku, Kawasaki
211, Japan
Tel: +81-44-754-2659 Fax: +81-44-754-2582
E-Mail: hirata@lucy.stars.flab.fujitsu.co.jp

KEY WORDS AND PHRASES

Force-reflection, peg-in-hole, predictive display, teleoperation, virtual collision.

ABSTRACT

We developed a force-reflecting teleoperation system that uses a real-time graphic simulator. This system eliminates the effects of communication time delays in remote robot manipulation. The simulator provides the operator with predictive display and feedback of computed contact forces through a six-degree of freedom (6-DOF) master arm on a real-time basis. With this system, peg-in-hole tasks involving round-trip communication time delays of up to a few seconds were performed at three support levels: a real image alone, a predictive display with a real image, and a real-time graphic simulator with computed-contact-force reflection and a predictive display. The experimental results indicate the best teleoperation efficiency was achieved by using the force-reflecting simulator with two images. The shortest work time, lowest sensor maximum, and a 100% success rate were obtained. These results demonstrate the effectiveness of simulated-force-reflecting teleoperation efficiency.

INTRODUCTION

In order to establish on-orbit manipulation and rendezvous-docking technologies, a satellite mounted with a robot manipulator will be launched in 1997[1]. The experiments involve a challenging attempt of master-slave teleoperation from the ground, which raises the crucial problem of communication time delay between the onboard system and the ground system. It is expected that a delay of 2 to 4 second will exit in each way, and this delay deteriorates teleoperation efficiency.

In the past, the following methods have been proposed to overcome this delay. The most primitive is the move-and-wait strategy, which is time-consuming and increases the fatigue of the operator. The predictive display provides a delay-free clear picture through a predictive simulator on a real-time basis. Even with its support, however, the operator tends to make large operational commands to the robot due to lack of contact force feedback. This situation can cause damage to the equipment or generate vibrations that affect the satellite's attitude. The compliance control of the slave arm is able to accommodate the force that is generated by excessive operational command input, but cause of the limited capacity of computer resources mounted on the satellite, damage or negative affects still occur. The bilateral master-slave manipulation loop is known to be unstable in teleoperations involving time delays above 1 second [5].

In this paper we propose computed-force-reflecting teleoperation using a real-time simulation and show its effectiveness through a typical teleoperation task of peg-in-hole. Originally, a similar idea was proposed in [4], [5], [6], [7], and [8], but the proposals did not include precise evaluations of the idea. We developed a teleoperation system including a display of degraded real images with a time delay, a real-time graphic simulator that provides contact force information, and a predictive display [3]. This enabled us to compare different types of teleoperation in practical basis. In this paper, we also propose a new concept of "virtual collisions in a virtual world". Based on this concept, the constraint force is generated from virtual objects that do not exist in reality. This force guides the slave arm along a safe path and prevents it from colliding with obstacles. With this system, high-precision peg-in-hole tasks involving round-trip communication time delays of up to a few seconds were performed at three support levels: a real image alone, a predictive display with the real image, and a real-time graphic simulator with computed-contact-force reflection and a predictive display. We show the experimental results which demonstrate the effectiveness of simulated-force-reflecting teleoperation.

COLLISIONS IN A VIRTUAL WORLD

In this section, we describe the concept of our teleoperation system. Real collisions and virtual collisions are implemented in the force-reflecting real-time simulator.

Real Collisions in a Virtual World

Real collisions in a virtual world involve collisions of similar objects built in a virtual world as well as the real world. For example, when collisions are generated between the slave arm and the equipment in a virtual world, they would also produce collisions in the real world. We define these collisions as "real collisions in a virtual world". Systems that can feed back the collision forces simulated by these models have

been reported [5], [6], [7], [8].

Because direct collisions between objects would produce collisions in the real world as well, they may cause equipment damage or generate vibrations that affect the satellite's attitude. To prevent damage and vibration and also to guide the slave arm to a safe position, we propose the new concept : Virtual collisions in a virtual world.

Virtual Collisions in a Virtual World

"Virtual collisions in a virtual world" are collisions between virtual objects that do not exist in reality. Collisions between virtual objects would not produce collisions in the real world because they do not exist in the real world. If this concept is applied to the master-slave operation, constraint force is generated from the virtual objects. It works to lead the slave arm to a safe position and to avoid direct contact with objects. We can create virtual objects of any kind in a virtual world, therefore, we can define a variety of constraint environments that do not exit in reality.

SOFTWARE IMPLEMENTATION

This section reviews the concept of real collisions and then discusses the generation of a constraint force from virtual collisions.

Implementation of Real Collisions in a Virtual World

A collision generated between the slave arm and an object on the satellite is noted as a real collision . It would also produce a collision in a virtual world, because similar objects are built in the virtual world. The force is calculated as a simple spring-loaded model as in the following equation (1).

$$\mathbf{F_b} = \mathbf{K}\,\Delta\mathbf{r} \qquad (1)$$

where **Fb** is the force of collision as viewed from the slave arm base coordinate system Σ **b**. **K** is a stiffness. Δ **r** denotes the distance between the surface and the current position of the tip of modeled slave arm.

Implementation of Virtual Collisions

To generate a constraint force needed to move the slave arm to a target position (see **Figure 1**), a virtual collision is considered. No collision is induced in the real world. Our 5-step method to generate a constraint force is as follows:

[Step 1] Calculate the collision force **Fb** at the virtual collision point as viewed from the slave arm base coordinate system.

"Collision point virtual frames" are coordinate systems that can be set to any point of the object collision. They are set at the each collision point when a virtual collision occurs between the slave arm and the virtual object (see **Figure 1**). There are twice as many as virtual frames as there are collision points.

[Step 2] Convert the value of **Fb** to **Fv**, the force of a collision point virtual frame.

$$\mathbf{F_v} = {}^{v}\mathbf{R_b}\,\mathbf{F_b} \qquad (2)$$

where ${}^{v}\mathbf{R_b}$ is a transformation matrix from the slave arm coordinate system to the collision point virtual frame.

[Step 3] Set the force sensor coordinate system, Σ **c**, on any position on the slave arm. This coordinate position corresponds to a position on the handle of the master arm.

[Step 4] Apply a virtual collision force, **Fv**, to the collision point virtual frame relative to the virtual collision point on the slave arm. Calculate the force, **fc**, and the moment, **nc**, by using the real-time simulator function of inverse dynamics calculation processing [3].

[Step 5] Generate the constraint force on the master arm side to facilitate operation.

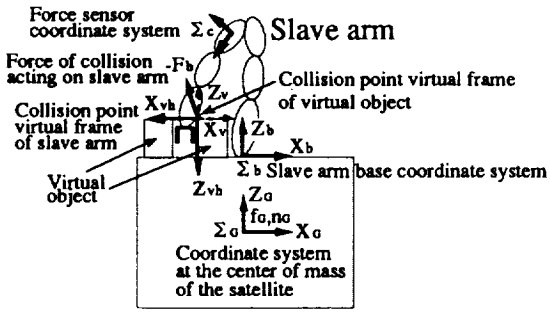


Figure 1. Virtual collision

Application for a Peg-in-Hole Task

Virtual Collision Model for a Peg-in-Hole Task Virtual collision forms vary according to the task nature. In a peg-in-hole task application, for example, proper positioning during movement to the vicinity of the hole should permit inserting the peg into the hole through constrained movement only in the z-axis direction. Safe, efficient, and reliable positioning is ensured by the use of a virtual collision model as shown in **Figure 2.** In this model, a virtual wire is set at the tip of the peg, a variable virtual column is set at the center of the hole opening, and a virtual plane is set around the hole.

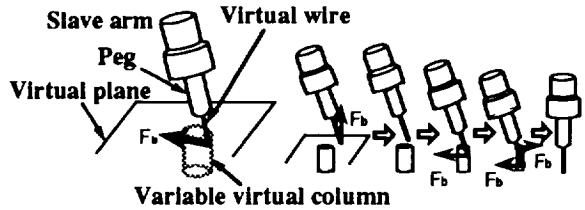


Figure 2. Virtual collision model

The model is characterized by :

(1) A real-time collision calculation resulting from the use of a virtual wire and a variable virtual column.

(2) Responsiveness to changes in the attitude of the tip of the slave arm derived from the use of a virtual wire.

(3) Reduced radius of the variable virtual column upon insertion of the virtual wire. As a results, a constraint force is generated to erect the virtual wire. A second order function is used as a radius function.

Constraint Force Calculation Processing The following sections describe the procedure for calculating the constraint force to be fed back to the operator through the master arm.

[Step 1] Calculate the constraint force **Fb** as viewed from the slave arm base coordinate system Σ **b**. This virtual collision case is shown in **Figure 3**.

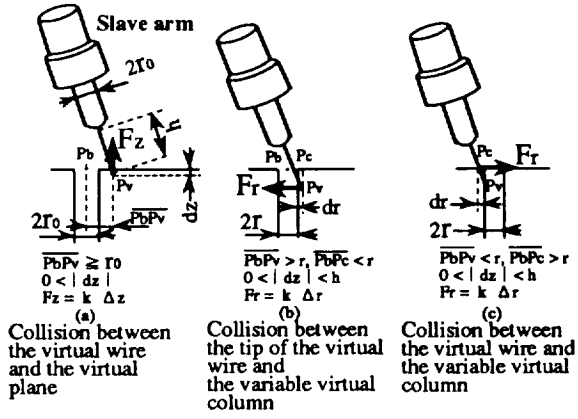


Figure 3. Virtual wire collisions

[Step 2] Calculate the force, **fc**, and the moment, **nc**, acting upon the force sensor coordinate system as mentioned in the above section. Feed back this force and moment to the master arm on a real-time basis. This force enables the operator to reach the center of the hole opening quickly and safely. An operator can sense the constraint force through the master arm. A damping term is attached to the force **Fb** to stabilize the transition to a non-collision state [7].

$$\mathbf{F_b} = \mathbf{K}\,\Delta\mathbf{r} + \mathbf{D}\dot{\Delta\mathbf{r}} \qquad (3)$$

where **K** is a stiffness, **D** is a damping coefficient, and Δ**r** denotes the distance between the surface of the virtual object and the current position of the virtual wire.

EXPERIMENT

Peg-in-Hole Tasks with Three Teleoperation System

Peg-in-hole tasks involving round-trip communication time delays were performed at three support levels: a real image alone, a predictive display with the real image, and a real-time graphic simulator with computed-contact-force reflection and a predictive display. The round-trip communication time delays used were 0, 4, and 8 seconds. The peg was 30 mm in diameter. The clearance between the peg and the hole was 0.9 mm. The depth of hole was 20 mm. The slave arm controller had local compliant control to avoid damaging the equipment.

Teleoperation System Configuration

Our system configuration is shown in **Figure 4**. The operator traces a path through the 6-DOF master arm while viewing the slave arm on a real image or predictive display. Simulated collision force is fed back to the operator through the master arm. The real slave arm follows the created path with time delays. **Figure 5** shows a man-machine interface system. The monitor display, force-reflecting simulator display, and master arm are arranged from left to right.

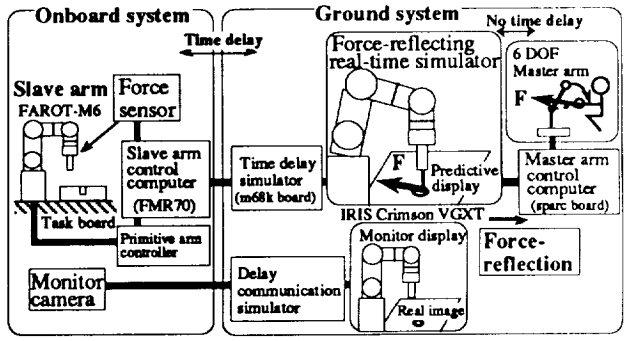


Figure 4. System configuration

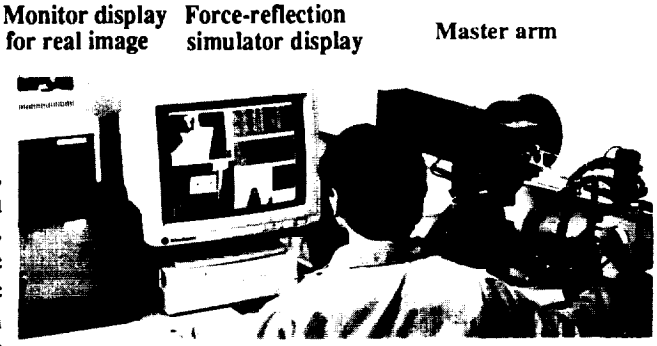


Figure 5. Man-machine interface system

RESULTS

The experimental results are indicated below. **Figure 6** is a diagram of task time when using a real image. Longer time delays prolong the completion times. For an 8-second time delay, an operator supported by a real image alone could not perform "move-and-wait" type teleoperation.

We compared the support levels of teleoperation with round-trip time delay of 4 seconds. **Figure 7** is a diagram of the total time of each task. The task time for the simulated-force-reflecting teleoperation is the shortest. The constraint force reduces the time needed to move to the vicinity of the hole. The time of peg insertion was not affected by the teleoperation support levels, because peg insertion was carried out through constrained movement in the z-axis direction. **Figure 8** is a diagram of force sensor maximums. The sensor maximum for the simulated-force-reflecting teleoperation is also the lowest. **Figure 9** shows the sample records of force sensor measurements. For both the real image and the predictive display, large amplitude and vibration were measured, while for the simulated-force-reflecting teleoperation, the measurements varied much less. **Table 1** summarizes the effects of support levels and lists the success rates. We regarded the result as a successful execution when the peg was inserted into the hole. The success rate was calculated from several trials. The success rate

of using the force-reflecting simulator is 100%. Overall, the simulated-force-reflecting teleoperation returned the best performance. The force-reflecting simulator provided us essentially identical performance even for 8-second time delays.

CONCLUSION

This study demonstrates the effectiveness of simulated-force-reflecting teleoperation. The experimental results with peg-in-hole tasks indicates the best teleoperation efficiency was provided by the force-reflecting simulator. The results also demonstrate the effectiveness of teleoperation based on the concept of virtual collisions in a virtual world. Feedback of a constraint force from virtual objects results in safer, more efficient, and more reliable task execution. We plan to apply these new teleoperation concepts to such tasks as paddle expansion and screw tightening.

REFERENCES

[1] M.Oda, Y.Wakabayashi, S.Ichikawa, R.Imai, and T.Anzai, 1992. ETS-VII, The World First Telerobotic Satellite (Mission And Its Design Concept),*Artificial Intelligence, Robotics And Automation, In Space*,307-318.

[2] Whitney,D.E., 1985. Historical Perspective and State of the Art in Robot Force Control, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 262-268.

[3] F.Nagashima,Y.Nakamura,1992. Efficient Computation Scheme for the Kinetic and Inverse Dynamics of a Satellite-Based Manipulator, *Proceeding of the 1992 IEEE International Conference on Robotics and Automation*, vol.1, 905-912.

[4] Richard Paul, Thomas Lindsay, and Craig Sayers, 1992. Time Delay Insensitive Teleoperation, *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 247-254.

[5] W.S.Kim, B.Hannaford, and A.K.Bejczy, 1992. Force-Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay, *IEEE Trans. on Robotics and Automation*, vol.8, no.2, 176-185.

[6] T.Kotoku, K.Tanie, and A.Fujikawa, 1990. Force-Reflecting Bilateral Master-Slave Teleoperation System in Virtual Environment, *Proceedings of the i-SAIRAS '90*, 295-298.

[7] T.Kotoku, 1992. A Predictive Display with Force Feedback and its Application to Remote Manipulation System with Transmission Time Delay, *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.1, 239-246.

[8] S.Tachi, H.Arai, T.Maeda,1990. Tele-existence master slave system for remote manipulation (II), *Proceedings of the 29th IEEE Conference on Decision and Control*, Vol.1, 85-90.

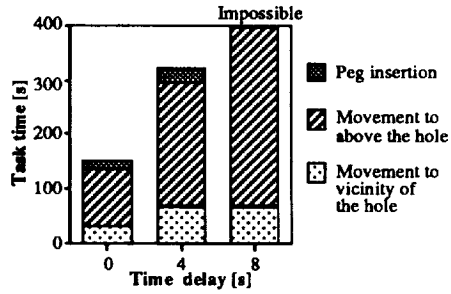


Figure 6. Task time with real image support

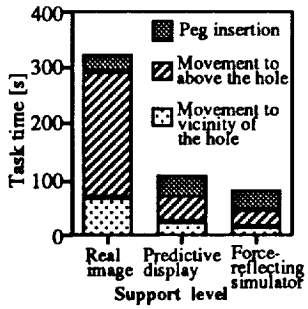


Figure 7. Task time
(Time delay : 4 [s])

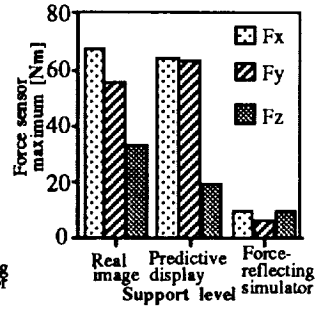
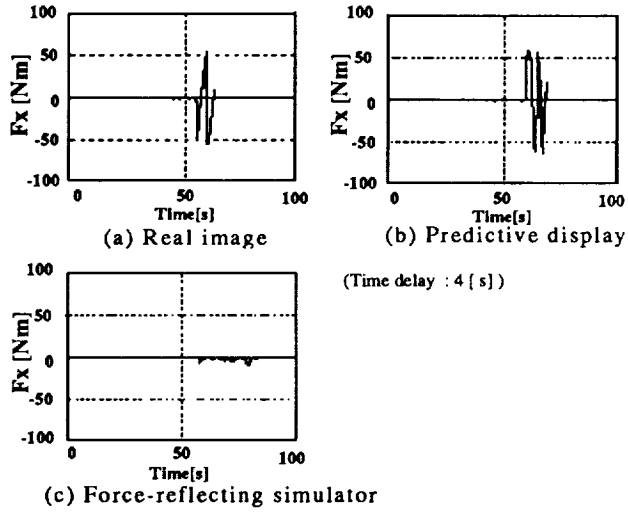


Figure 8.
Force sensor maximum
(Time delay : 4 [s])



(Time delay : 4 [s])

Figure 9. Sample records of force sensor measurements

Table 1 Comparison of the teleoperation support levels

Support level	Time delay [s]	Task time [s]				force sensor maximum |Fx|(N) |Fy|(N) |Fz|(N)	Success rate [%]
		Movement to above the hole	Movement to vicinity of the hole	Peg insertion	Total		
Real image [2]	4	68	225	29	322	67 55 33	38
Predictive display [3]	4	25	45	35	105	64 63 19	80
Force-reflecting simulator [3]	4	18	30	33	81	9.9 6.7 9.6	100
Force-reflecting simulator	8	14	30	32	76	6.8 3.6 10.0	100

1 Round-trip
2 Resolution:352 by 240,gray levels:16, frame rate:10 frames/s
3 Frame rate: 8 frames/s IRIS CrimsonVGXT (85MIPS,16MFLOPS,180kPolygon/s)